





DETERMINING WHETHER THE CONTENTS OF A DATA FILE  
ARE SUFFICIENT TO FILL A RETRIEVAL REQUEST

Christopher R. Sprague

March 1970

449-70



3 2 70  
MARY

DETERMINING WHETHER THE CONTENTS OF A DATA FILE  
ARE SUFFICIENT TO FILL A RETRIEVAL REQUEST

Christopher R. Sprague

March 1970

449-70

This paper is a direct extension of ideas developed in my S.M. Thesis.  
I am indebted to J. C. Emery and D. C. Carroll for their aid.



We present a solution to a problem of considerable practical interest which arises in the design of Management Information Systems. After presenting an example of the problem, we shall formalize it.

Example: Suppose we have a personnel file containing complete data on all our 20,000 workers. The Operations Research Group, wishing to conduct a study of absenteeism among older workers, has created a much smaller file containing the records of all those aged 60 or over as well as those of any age who were absent 10 or more days in the last year. Let us suppose that this file is 1500 records long.

The personnel department now wishes to obtain the names of all employees within 2 years of compulsory retirement at age 65. The desired list can be obtained either from the large file owned by the personnel department or from the smaller file owned by the Operations Research Group. Question: How would the information system "know" that the list could be obtained from the shorter file (at some considerable saving)?

Formalization: Let a "record" be a collection of integers  $k_1, \dots, k_j, \dots, k_u$ . In general, the value of  $k_j$  for any record implies nothing whatsoever about the value of  $k_j$  for another record. Let a "file" be a collection of  $m$  records. Let there be a special file called the "master file" which consists of all records of interest.

Let a "request"  $R_i$  be a function which maps the integer contents of a record into the values "true" and "false". Let a "strip file"  $S_i$  be a file produced as follows: The master file is read record by record. As each



record is read, the request  $R_i$  is applied to the integer contents of that record. If and only if the value of  $R_i$  is "true", the record is copied to the strip file  $S_i$ .

Further, let the data base at any moment consist of the master file and  $p$  strip files  $S_1, \dots, S_i, \dots, S_p$ , produced from requests  $R_1, \dots, R_i, \dots, R_p$ , respectively.

Now, a new request  $R_{p+1}$  is entered into the system. If another request, say  $R_a$ , can be found such that  $R_{p+1}$  implies  $R_a$ , then it must be true that all of the records in the master file which would cause  $R_{p+1}$  to be true are also in  $S_a$ . In other words, the records in  $S_{p+1}$  will be a subset of the records in  $S_a$ . In this case, there is probably an advantage in creating  $S_{p+1}$  from  $S_a$  rather than from the master file.

Procedure: Determining whether some request  $R_c$  implies some other request  $R_d$  can be accomplished by a technique called "recursive residual reduction".<sup>1</sup> First let us consider requests formed in terms of logical variables (i.e., those taking on only the values "true" and "false" (abbreviated as 1 and 0 respectively); three Boolean operators \* ("and"), + ("inclusive or"), and ' ("not"); and parentheses; all with their customary meanings. (We assume that \* takes precedence over + in the hierarchy of operators.)

We will use the capital letters A-H as logical variables. Consider two requests:

$$R_1 = A+B$$

$$R_2 = A*(C+B)'$$



$R_1$  is true if either A or B (or both) is true, and  $R_2$  is true if A is true, but neither B nor C is true.  $R_2$  implies  $R_1$ , since  $R_1$  is always true if  $R_2$  is true. This fact can be determined by a simple procedure, based on the "Expansion Theorem"<sup>2</sup>, which states that  $f(x_1, \dots, x_i, \dots, x_n) = x_i * f(x_1, \dots, 1, \dots, x_n)$   $x_i' * f(x_1, \dots, 0, \dots, x_n)$  where f is any Boolean expression in logical variables  $x_1, \dots, 1, \dots, x_n$ . The expressions  $f(x_1, \dots, 1, \dots, x_n)$  and  $f(x_1, \dots, 0, \dots, x_n)$  are called the "residue about  $x_i$ " and the "residue about  $x_i'$ " respectively. Implication can be determined by the following recursive procedure:

Let U and V be two Boolean expressions.

1. U does not imply V if  $U = 1$  and  $V = 0$ .

2. U implies V is  $U = 0$  or  $V = 1$  or both.

3. If neither rule 1 nor rule 2 applies, choose a logical variable, say Q, appearing in either U, V, or both. (Call this variable the "pivot variable".) Take residues of U and V about Q and  $Q'$ . U implies V if and only if the residue of U about Q implies the residue of V about Q and the residue of U about  $Q'$  implies the residue of V about  $Q'$ .

Reducing a residue to simpler form is achieved by applying the following rules: Let E be any logical expression (i.e., a logical variable, a constant 0 or 1, or an expression enclosed in parentheses).

a)  $1 + E = E + 1 = 1$

b)  $1 * E = E * 1 = E$

c)  $0 + E = E = 0 = E$

d)  $0 * E = E * 0 = 0$

e)  $0' = 1$

f)  $1' = 0$



g) parentheses should be removed where not needed for operations hierarchy, i.e., from any single logical variable or constant; or from a complete expression where neither preceded nor followed by an operator.

Let us go back to  $R_1 = A + B$

$$R_2 = A*(C+B)'$$

and apply our rules to see whether  $R_2$  implies  $R_1$ .

First, we see that neither rule 1 nor rule 2 applies directly, so invoking rule 3 and choosing A as pivot variable, we have

	Residue about A	Residue about A'
$R_1$	$1 + B = 1$	$0 + B = B$
$R_2$	$1 * (C+B)' = (C+B)'$	$0 * (C+B)' = 0$

By rule 2, it will be seen that the residue of  $R_2$  about A implies the residue of  $R_1$  about A (because the residue of  $R_1$  about A is 1); and the residue of  $R_2$  about A' implies the residue of  $R_1$  about A' (because the residue of  $R_2$  about A' is 0). Therefore  $R_2$  implies  $R_1$ .

In practice we would probably not represent the logical expressions symbolically, but rather as a "Polish" string of operators and operands with all negations resolved by De Morgan's theorem.<sup>3</sup>

Now let us return to our original problem of two personnel files. We must extend our language to include the relational operators  $=$ ,  $\neq$ ,  $>$ ,  $\geq$ ,  $<$ ,  $\leq$ ,  $\cdot$ . Suppose that integers  $k_i$  and  $k_j$  represent age as of January 1 and days absent last year respectively. The file owned by the Operations Research Group was produced by the request  $R_3 = (k_i \geq 60) + (k_j \geq 10)$ . The file desired



by the personnel department could be described by  $R_4 = (k_i \geq 63)$ . Our first step is to create logical variables corresponding to each relational operation. Let:

$$A = (k_i \geq 60)$$

$$B = (k_j \geq 10)$$

$$C = (k_i \geq 63).$$

That we are permitted to do this is clear, since a relational operation is, for any record, either true or false. Our requests are now

$$R_3 = A + B$$

$$R_4 = C$$

But we know that both A and C are derived from the same integer, namely,  $k_i$ . In fact, if C is true, A must also be true. Therefore, C implies A. Recognizing this fact makes us treat the Expansion Theorem more generally: Any residue about C will substitute 1 for both A and C. Any residue about A' will substitute 0 for both A and C.

The reasoning behind this is clear. The residue of U about X is simply the form of U, remaining if X is assumed true. But if X implies Y, then the residue of U about X must also assume Y to be true. Similarly, the residue of U about Y' must assume X to be false.

Let us now try the generalized procedure on  $R_3$  and  $R_4$ , remembering that C implies A.

$$R_3 \quad A + B$$

$$R_4 \quad C$$

Neither rule 1 nor rule 2 applies so applying rule 3 and choosing B as pivot variable, we have



	Residue about B	Residue about B'
R <sub>3</sub>	A + 1 = 1	A + 0 = A
R <sub>4</sub>	C	C

The residue of R<sub>4</sub> about B implies the residue of R<sub>3</sub> about B by rule 2, but neither rule 1 nor rule 2 applies to the residues about B', so we invoke rule 3, choosing C as pivot variable:

	Residue about C	Residue about C'
Residue of R <sub>3</sub> about B'	A = 1 (by C implies A)	A
Residue of R <sub>4</sub> about B'	1	0

Both residues of R<sub>4</sub> imply their corresponding residues of R<sub>3</sub>, therefore R<sub>4</sub> implies R<sub>3</sub>.

Extensions: The general procedure described herein is capable of a great many extensions:

1. File structure -- Our definition of a master file is phrased so that it includes files stored on direct-access devices as well as sequential devices. The notion of a strip file is easily generalized then to a list of those records stored in a direct-access master file which meet some criterion (request).

2. Other Relationships -- Equivalence, negation, contraindication, etc., can be determined by simple modifications of the procedure described here.<sup>4</sup>

3. Other Operations -- All logical functions of two variables (there are at most 16) can be easily handled by appropriate additions to rules a) - g). Moreover, one practical situation which we have recently faced required the definition of n-ary equivalents of "or" and "and".<sup>5</sup> Extension of this technique to n-ary operations is straight forward.



4. Finding the shortest qualifying strip file -- If we order strip files  $S_1 \dots S_p$  in ascending order of length, then iteratively apply this technique to each  $R_i$  in turn with respect to  $R_{p+1}$ , the first  $R_i$  implied by  $R_{p+1}$  is the shortest qualifying file. If none qualify, the master file must be used.

#### Conclusion

The "Expansion Theorem" is a powerful tool in the analysis of the contents of data files.



REFERENCES

1. See Sprague, C.R., "On Efficient Multiple-User Date Retrieval From Serial Files", unpublished S.M. Thesis, M.I.T., 1964.
2. S.H. Caldwell, Switching Circuits and Logical Design, Wiley, New York, 1958, Chapter III, Theorem 20.
3. Sprague, op. cit.
4. Sprague, op. cit., Chapter IV. Section B.
5. A paper is forthcoming.





2-27-64

Date Due

FEB 05 '78 JUL 6 '78		
APR 02 '77		Lib-26-67

MIT LIBRARIES

DUPL



441-70

3 9080 003 702 237

MIT LIBRARIES

DUPL



443-70

3 9080 003 702 211

MIT LIBRARIES

DUPL



444-70A

3 9080 003 671 200

MIT LIBRARIES

DUPL



445-70

3 9080 003 671 242



446-70

3 9080 003 671 218

MIT LIBRARIES

DUPL



447-70

3 9080 003 671 358

MIT LIBRARIES

DUPL



448-70

3 9080 003 702 344

MIT LIBRARIES

DUPL



449-70

3 9080 003 702 310

MIT LIBRARIES

DUPL



450-70

3 9080 003 671 309

MIT LIBRARIES

DUPL



451-70

3 9080 003 671 325

MIT LIBRARIES

DUPL



452-70

3 9080 003 671 333

